REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED Final
	January 2002	~~January 1998 - 30 December 2002~~ 01 Apr 98 - 30-Sep 02

4. TITLE AND SUBTITLE
Optimization Problems in High-Speed Networks

5. FUNDING NUMBERS
DAAG55-98-1-0170

6. AUTHOR(S)
Serge Plotkin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Stanford University
Department of Computer Science
Stanford, CA 94309

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
37581-MA
.1

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The main goal of this project is to develop fundamental algorithmic techniques that can be applied to problems that arise in the context of high-speed communication networks. The explosion in the size and complexity of networks, together with QoS requirements needed by some of the new services raises many new challenging problems. In this report we outline our latest accomplishments in the areas of online routing of FTP requests and snapshot scan algorithms.

14. SUBJECT TERMS
High-speed networks, quality of service, network topology, algorithmic design

15. NUMBER OF PAGES
17

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Final Report

PERIOD COVERED BY REPORT:	January 1, 1998 – December 31, 2002
GRANT NUMBER:	DAAG55-98-1-0170
NAME OF THE INSTITUTION:	Stanford University
PROJECT TITLE:	Optimization problems in High-Speed Networks
PI:	Serge Plotkin Dept. of Computer Science, Stanford University (415)-723-0540 plotkin@cs.stanford.edu
STUDENTS SUPPORTED BY THE CONTRACT:	Kamesh Munagala, Adam Meyerson, Ashish Goel, Zoe Abrams
PHD DEGREES GRANTED:	3
MSC DEGREES GRANTED:	1

1 Research Objectives and Motivation

The main goal of the project is to develop fundamental algorithmic techniques that can be applied to problems that arise in the context of high-speed communication networks. The explosion in the size and complexity of networks, together with QoS requirements needed by some of the new services raises many new challenging problems.

Network topology design is one of the basic problems faced by companies that are trying to build a new infrastructure or to expand the existing one. New tools are needed in order to decide, in a principled way, on where to put the "concentrators", which communication links to purchase/lease, and which communication equipment to buy. Essentially all of the problems in this class are NP-hard and hence we have to design approximation algorithms. The goal is to design *efficient* algorithms that have provable performance guarantees (i.e. approximation factors). Here by "efficient" we do not mean just "polynomial time". Since the size of input is usually very large, it is important to design algorithms that run in nearly linear time.

Once the network topology is designed, the next step is to decide where to place servers and caches. Server placement should take into account the topology of the network and should minimize the delay experienced by the customers. Distributing (and, possibly, replicating) applications and data among the servers should take into account the load placed on the servers. The goal is to minimize the total expense of purchasing, installing, and maintaining the servers.

Since the resources (e.g. bandwidth) in a communication network are limited, it is important to divide them between users in a *fair* way. In case of bandwidth, the standard fairness definition is max-min fairness. Roughly speaking, bandwidth allocation is considered fair if "poor" connections can not increase their bandwidth by stealing from "rich" connections. We showed that this definition is not applicable to the case where it is possible to choose different routes to different types of connections and worked on several algorithms that achieve approximately fair routing and bandwidth allocation in the online setting.

2 Results

Network Topology Design A significant part of the cost of building a new network infrastructure is in purchasing/leasing the communication links and the associated communication equipment. Due to economies of scale, the cost structure is usually *concave*. For example, installing a DS-3 link is usually much cheaper than installing 30 separate T1 links and using a reverse multiplexer; installing an OC-3 link is usually cheaper than purchasing 3 separate DS-3 links. In general, the higher the bandwidth of the link, the cheaper is the cost per megabit per second when we take into account both the cost of the link and the cost of the communication equipment at the endpoints of the link.

The concave cost structure prevents us from applying classical flow techniques directly. This is in contrast to the convex cost case, where the problem can be reduced to min-cost flow (or, in some cases, multicommodity flow) by approximating each link with several linear-

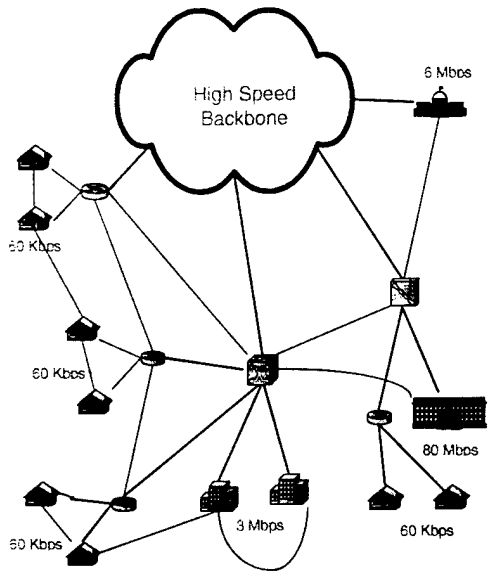

Figure 1: Buy-at-bulk input.

cost capacitated links. It is easy to see that even the simplest problems with concave cost structure are NP-hard. For example, consider the simplest case where link costs are identical and independent of capacity/bandwidth (i.e. fixed-cost infinite capacity links). The problem in this case is exactly the Steiner Tree problem. In other words, Steiner Tree problem can be stated as finding an uncapacitated min-cost flow from several sources to a sink where the cost is a step function, i.e. cost is zero for zero flow and cost is equal 1 for flow above 0.

The *buy-at-bulk* network design problem was defined in [SCRS97]. Roughly speaking the problem is as follows: Given a graph with concave costs on the edges and a list of source/sink pairs with bandwidth demands, construct the cheapest network that will support all of the demands. Awerbuch and Azar [AA97], gave an $O(\log^2 n)$ randomized approximation algorithm to the buy-at-bulk problem. Their algorithm was based on the randomized tree construction of Bartal [Bar96]. This construction was later improved by Bartal to give $O(\log n \log\log n)$ bound [Bar98]. Using the new construction, it is easy to prove that the algorithm in [AA97] gives a better bound as well. We have shown how to derandomize the latter construction in [CCG+98], while maintaining the performance bound.

The main deficiency of the Awerbuch/Azar approach is that it requires that the *cost functions on all the edges are the same.* Since most networks are not designed "from scratch", different possible links may have different cost structures, for example depending on the already available "dark fiber" or pre-installed copper. Moreover, uniform cost requirement means that we can not solve problems where we have to both design an inexpensive network and to locate the servers at the same time. (Observe that a server can be represented as a special edge with a different cost function.)

Together with Meyerson and Munagala (PhD students supported by the Contract), we have

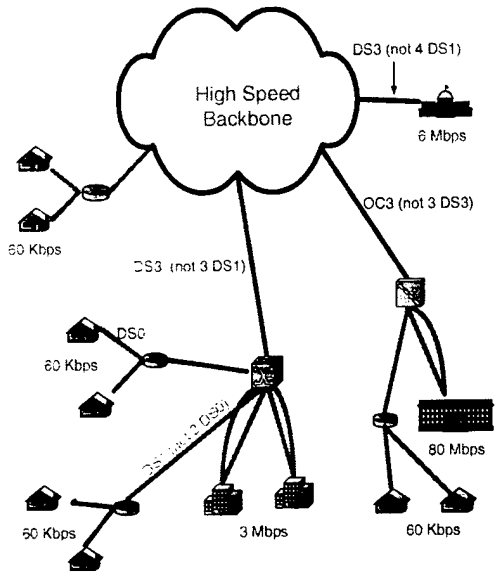

Figure 2: Buy-at-bulk output.

considered the buy-at-bulk problem where the costs of different edges can be different. The main result is a simple $O(\log n)$ approximation algorithm for the case where there is only one source [MMP00].

In fact, we solved a more general problem that can be used as a framework to solve a large class of network design and server placement problems: Given a graph where each edge has two different costs $\sigma(e)$ and $\ell(e)$ and a list of sources $\{s_i\}$ with demands $\{d_i\}$, the goal is to minimize the following expression:

$$\sum_{e \in T} \sigma(e) + \sum_i d_i \text{dist}_\ell(r, s_i)$$

Intuitively, $\sigma(e)$ corresponds to the fixed cost of purchasing a communication link, and $\ell(e)$ corresponds to the incremental cost of using more bandwidth along this link.

In a large class of problems that appear in practice the fixed and incremental costs are not arbitrary. Roughly speaking, it is usually not useful to purchase a high-capacity (and high fixed-cost) link unless we can use at least a constant fraction of the capacity of this link. Formally, these problems fall under the "access networks" class.

Meyerson and Munagala worked on the design of access networks and, together with Guha, showed an $O(1)$ approximation algorithm in [GMM00]. The also extended the approach to give a constant approximation algorithm to the more general single-source buy-at-bulk problem [GMM01b]. In [MMP01a] we show how to extend this approach to obtain the first constant factor approximation to the online version of this problem, where the "customers" appear online in a random order.

We have implemented the above algorithms in order to evaluate their performance. Figure 3

n	k	Our Alg 1, %err	Our Alg 2, %err	Alg 2 time	LP time
18	4	7.69%	11.12%	1.6s	3.78s
20	4	6.13%	2.62%	2.04s	7.73s
30	4	0%	13.98%	6.20s	229.83s
50	3	8.54%	11.24 %	31.16s	1842.8s

Figure 3: Access network design - preliminary results

shows some of the results. In this figure, n is the number of nodes in the graph and k is the number of different "pipes" along each link, i.e. $k-1$ is the number of breakpoints in the piece-wise linear approximation to the concave cost. The underlying graph was created by taking locations of 180 US cities, choosing a random subset of 18 to 50 nodes, and building a random graph on these nodes. The lengths were taken as real distances between the cities, and we assumed that we can purchase DS0, DS1, DS3, or OC3 along each link. (For simplification, we did not consider DS0 for the 50-node graph.)

Note that our approaches produce results within 15% of optimum fractional solution. The important issue is scalability. As we can see from the figure, the running time of the Linear Programming solver (we used CPLEX) increases dramatically with the size of the graph, much faster than the increase in the running time of our algorithms.

Network Design and Multicommodity Flow. In [GOPS97, GOPS98] we describe implementation results of a combinatorial approximation algorithm for the *minimum-cost multicommodity flow problem.* This problem involves simultaneously shipping multiple commodities through a single network so that the total flow obeys arc capacity constraints and has minimum cost. This problem often arises when one wants to design topology of a communication infrastructure and to determine capacities on the links.

Multicommodity flow problems can be expressed as linear programs, and most theoretical and practical algorithms use linear-programming algorithms specialized for the problems' structures. Combinatorial approximation algorithms in [GK95, KP95, PST95] yield flows with costs slightly larger than the minimum cost and use capacities slightly larger than the given capacities. Theoretically, the running times of these algorithms are much less than that of linear-programming-based algorithms.

We combined and modified the theoretical ideas in these approximation algorithms to yield a fast, practical implementation solving the minimum-cost multicommodity flow problem. Experimentally, the algorithm solved our problem instances (to 1% accuracy) two to three orders of magnitude faster than the linear-programming package CPLEX and the linear-programming based multicommodity flow program PPRN [CN96].

Online multicast. In [GHP98] we presented an online algorithm for construction of multicast trees. This paper considered the model where, at any instance in time, a user (residing in one of the nodes of the given communication network) requests to join an existing multicast group

or to start a new group. The routing protocol should either *reject* the request or *accept* it and allocate the requested bandwidth along a path connecting the new endpoint with the already existing tree for this group. Without loss of generality we can assume that the protocol always accepts requests to create a new group. The goal is to satisfy as many requests as possible.

The above publication presents the first polylog-competitive algorithm for the general multicast problem. Our algorithm is randomized since it is impossible to achieve polylog competitive ratio by a deterministic algorithm. The ratio of the number of requests accepted by the optimum offline algorithm to the expected number of requests accepted by our algorithm is $O(\log \mathcal{M}(\log n + \log\log \mathcal{M}) \log n)$, where $\mathcal{M}$ is the number of multicast groups and n is the number of nodes in the graph. If each vertex is allowed to serve at most one multicast group, the competitive ratio simplifies to $O(\log^3 n)$.

A natural question to ask is if it is possible to make the competitive ratio independent of $\mathcal{M}$, the number of multicast group. We address this issue by showing a lower bound of $\Omega(\log \mathcal{M} \log n)$ when $\mathcal{M}$ is much larger than the link capacities. This is the first bound for this problem that is stronger than $\Omega(\log n)$.

Approximation through tree metrics. Numerous NP-hard problems on general weighted graphs become much easier to approximate when we restrict our attention to weighted trees. In [Bar96, Bar98], Bartal gave a randomized polynomial time algorithm to construct a tree such that, in expected sense, the distances between any points on the tree are close to the distances in the original graph.

Using his algorithm, one can design approximation algorithms for a large class of optimization problems. Roughly speaking, we first build a tree that approximates the distances, solve the problem on this tree only (disregarding the rest of the edges), and reinterpret the resulting solution in terms of the original graph.

Since Bartal construction is randomized, all of the algorithms resulting from the above approach are randomized. In [CCG+98] we have shown how to *derandomize* Bartal's construction. More precisely, we give an efficient polynomial time algorithm that given a finite n point metric G, constructs $O(n \log n)$ trees and a probability distribution μ on them such that the expected stretch of any edge of G in a tree chosen according to μ is at most $O(\log n \log\log n)$. Our result establishes that finite metrics can be probabilistically approximated by a *small* number of tree metrics. In other words, we show that given an arbitrary n-point weighted graph, we can construct a *polynomial* number of trees such that if we pick one of these trees at random, the expected distance along this tree is at most a factor of $O(\log n \log\log n)$ more than the real distance in the original graph.

Our approach immediately gives deterministic algorithms for many optimization problem. In particular, it gives a deterministric approximation algorithm for the buy-at-bulk problem. In this problem we are given pairs of points and capacity demands between them. We are also given a graph that specifies where we can buy links and how much do we pay per link of certain capacity. The costs are assumed to be concave, i.e. buying two links of capacity c is *more expensive* than buying a single link of capacity $2c$. The goal is to construct the cheapest network

that satisfies the capacity demands. The only previously known approximation algorithm for this problem [AA97] was randomized.

Caching and Data Placement Web and application caches are essential part of the infrastructure. Proper use of data replication and caching can dramatically improve the response time of the system and, in general, the overall "user experience". Thus, it is important to develop principled tools to help choose where to place the caches, which data to place in which cache, and how to manage the replication policies.

Most theoretical work on web caching [CK99, AAK99, Ira97] has focused on online page replacement policies. The goal is to design a dynamic (online) policy that will be not "overflow" caches and and the same time will be able to satisfy most user requests for pages from nearby caches.

Recent experimental observations [BCF+99, ZIRO99, TRS97, CK99] tend to suggest that the domain demand vectors change at a relatively course time scale and hence it is useful to consider semi-static replication policies that assume statistical knowledge of data access patterns. More precisely, given access statistics for each data block, the goal is to distribute the data among the given set of servers (replicating some of the data) in order to minimize the average response time, measured as distance between the user and the page it is trying to access. This data distribution/replication problem was considered in [KPR99], who showed how to approximate within a constant factor the average user distance to a cache without overflowing it, under some assumptions on the distance (delay) metric.

In [MMP01b] we proposed to consider another parameter, which we called "user load". More precisely, we assumed that each server has two independent parameters: the total size of the pages that it can hold, and the total number of pages that it can serve per second (user load). Note that the size of the page does not necessarily refer to the number of bytes. For example, we can use page size to represent the amount of update traffic that will be caused by placing this page in the server.

Our algorithm in [MMP01b] distributes the data among the servers (replicating some of the data) without exceeding the page size and the load bound by more than $O(\log C)$ factor, where C is the number of servers. In addition, our algorithm makes sure that, in the worst case, the average user's network distance from his assigned cache is within a factor of 5 of the optimum. If we are not allowed to use replication, we can improve these bounds to exceed cache size and the user threshold by only a factor of 4 while obtaining optimum average distance-to-cache. We also showed that if we are allowed to choose where to place servers, we can obtain a constant approximation to the average delay experienced by users and total server cost without overflowing the caches and without exceeding the user bound.

The above mentioned approaches apply only to relatively small size files. New approaches are needed in order to cache *streaming media*. First, the size of some streams (e.g. video streams) can be extremely large which means that it is impossible to fit entire streams in a cache. Second, we are more concerned with the maximum number of simultaneous cache misses rather than the total number of cache misses, since each cache miss results in a flow of data

from the main server to the cache, increasing the load on the server-cash route.

Following dynamic caching framework (for a single cache and a single server) was considered in [DS96, HNG+99]. Suppose that there is a request for some data stream at time t_1 and another request for the same data stream at time $t_2 = t_1 + \Delta$. (In the terminology of [HNG+99], Δ is the *temporal distance* between the two requests.) Suppose also that there is enough space in the cache to store Δ time units of the stream. Then, we can serve both requests using only a single connection to the server by caching the last Δ time units of the stream that were seen by the first request. The second request can always obtain from the cache the current stream data that it needs.

Andrews and Munagala [AM00] (Munagala is a current PhD student) addressed this problem in the case where the requests arrive online, where the goal is to minimize the maximum bandwidth ever used by the server. They showed that unlike traditional online page replacement policies (where the competitive ratio depends on the size of the cache [FKL+91, ST85, MS91]), it is possible to obtain almost optimal performance in this model. They also generalized this approach to work for arbitrarily many caches and video clips in a network setting.

Routing and Fairness Part of the process of network design and management is division of limited resources between the users. One such resource is the available bandwidth. When dividing the bandwidth, we have to provide some *fairness guarantees*. At the same time we would like to optimize the *total bandwidth* allocated to the users (throughput). It is not hard to show that these two goals are often contradictory.

Accepted definition of fairness is the *max-min fairness* [BG92]. Roughly speaking, bandwidth allocation is considered fair according to this definition if "poor" sessions can not increase their bandwidth by stealing from "richer" sessions that share links with them. Since this definition assumes that the routes are given, the standard approach consists of two independent steps. The first step computes the routes, and the second step divides the bandwidth along these routes in a max-min fair way. In the past we were successful in developing algorithms that can be used to implement the first step with provable performance guarantees [AAP93, AAF+93]. Several efficient distributed algorithms that implement the second step are known as well [AMO96a, AMO96b, AS98, BFCAZ99].

Recent advances in routing technology allow us to aggregate many TCP/IP connections together (for example, by hashing on the source/destination addresses) and route each aggregated flow in a different way. Thus, we need an alternative definition of fairness that, in addition to allowing changes in bandwidth, *allows rerouting of different flows* onto alternative paths. Moreover, we have to take into account that, as long as the final step implements exact max-min fairness, it is impossible to approximate globally optimum total throughput. (Locally maximum throughput, i.e. maximum throughput without rerouting, can be approximated by distributed algorithms such as in [BBR97]).

The generalized definition should be equivalent to max-min fairness in the case where the routes are already known and no rerouting is allowed. Kleinberg et.al. [KRT99] proposed the following definition of fairness in a variable-route scenario. Compute a *bandwidth vector*

composed of the bandwidth allocated to flows, in increasing order. The ith coordinate is thus equal to the bandwidth of the flow receiving ith least bandwidth. Also define a *prefix vector* where the ith coordinate is equal to the sum of the bandwidths assigned to the i minimum flows (i.e. the sum of the first i coordinates from the previously computed vector). Note that the first coordinate of the prefix vector represents the bandwidth given to the "most starved" flow while the last coordinate is equal to the total throughput. An allocation with *lexicographically maximum* bandwidth vector is considered *globally fair*. It's not difficult to show that the allocation with the larger bandwidth vector also has the larger prefix vector. If routes are preassigned, the fairest possible allocation of bandwidths according to this definition is the max-min fair allocation.

Using the above definition of global fairness, we can restate our goal as follows: We need to design a bandwidth allocation policy which assigns routes and bandwidths so as to guarantee that throughput is within a small factor of optimal and each term of the bandwidth vector is within a small factor of the globally fair bandwidth vector.

Unfortunately, the above stated criterion does not always lead to reasonable allocation of bandwidth. There are cases where one can start with a solution which achieves close to optimal throughput and assigns every flow close to its globally fair bandwidth, and where the bandwidth assigned to many "starved" flows can be increased by a polynomial factor, while reducing the total throughput by less than a constant factor.

This indicates that we need to modify our goal. Instead of trying to approximate the throughput and the globally fair bandwidth vector, we would like to simultaneously approximate *all of the possible bandwidth vectors*. More precisely, the goal is to design an algorithm that assigns routes and bandwidths such that its corresponding bandwidth vector is coordinate-wise at least a $1/\gamma$ fraction of *any possible bandwidth vector*. A γ-competitive bandwidth allocation guarantees, for any b, that the number of connections which were assigned bandwidth $\geq b$ is not lower than the number of connections which were assigned bandwidth $\geq \gamma b$ *in any legal solution*. It's fairly straightforward to show that this results in a prefix vector that is also coordinate-wise at least the same γ fraction of any possible prefix vector. Thus a γ-competitive bandwidth allocation also achieves throughput which is within a γ factor of optimum.

As we have observed in [GMP01], another way to explain why the above definition of fairness is reasonable is to relate it to the notion of *majorization*, studied by Hardy, Littlewood and Polya [HLP29, HLP52]. Let x and y be two valid resource assignment vectors, where the ith coordinate determines the amount of some limited resource (not necessarily bandwidth) allocated to user i. We say that x *is majorized by* y if, for every k, the sum of k smallest components of x is not less than the sum of k smallest components of y. Intuitively, if the above holds, then x is "not worse, or "at least as fair" as y, since poor users with respect to x resource distribution get at least as much as the poor users with respect to y distribution.

The above intuition can be formalized by using the fact that if x is majorized by y then for any function f that is symmetric and concave in its arguments, we have $\sum_i g(x_i) \geq \sum_i g(y_i)$. Suppose there exists some utility function which measures the fairness of an allocation. It is easy to see that under any "natural" definition of fairness, such function has to be symmetric

in its arguments and concave. Thus, maximizing fairness is equivalent to maximizing some concave function (or minimizing some convex function). Hence, under any "natural" definition of fairness, the best solution will be the one that is majorized by all possible vectors describing the resource distribution.

Unfortunately a vector x that is majorized by all possible y vectors does not need to exist. For example, the set of constraints $\{x_1 + x_2 + x_3 = 6; 2x_1 + x_2 \leq 3; x_1, x_2, x_3 \geq 0\}$ does not admit such a vector. A natural approach is to relax the definition of majorization. We say that vector x represents an γ-fair solution if γx is majorized by all possible vectors representing resource distribution among the users. Intuitively, this means that the k poorest users together in a globally γ-fair solution must be at least $1/\gamma$ times as rich as the k poorest users in any other allocation.

In [GMP01] we described the relationship between majorization and fairness and considered the job assignment problem where each job can be assigned to a subset of machines. Here, the limited resource is the computation power of the machines and the ith coordinate of the resource assignment vector corresponds to the proportion of machine assigned to job i. We showed that if the jobs arrive online, the standard greedy algorithm is $O(\log n)$-fair.

In [GMP00] we presented an *online* algorithm that routes the connections and assigns bandwidth to each connection such that the resulting γ is $O(\log^2 n \log U)$, where n is the number of nodes in the network and U is the maximum number of connections routed by an optimum algorithm on a single link. Recently, *offline* variant of a closely related bandwidth assignment problem was considered in [KK00], who presented an $O(\log U)$ approximation algorithm for the case where the routes are fixed and all we need to do is decide how much bandwidth to allocate to each route.

Figure 4 compares a variant of our algorithm in [GMP00] (marked GMP) to algorithm which routes over random min-hop paths (marked Random) and to algorithm based on our earlier work [AAF+93] that routes along shortest paths with respect to cost that is exponential in the load(marked EXP). Experimental results show that the latter algorithm achieves good global fairness. The graph shows scaled bandwidth vector, i.e. we divide the i-th coordinate (total bandwidth assigned to the poorest i users) by i. Note that the leftmost point corresponds to the minimum bandwidth assigned to a connection, while the rightmost point corresponds to average bandwidth per connection, i.e. total throughput divided by the number of connections. Observe that our total throughput is significantly more than the throughput achieved by the other algorithms while our minimum assigned bandwidth does not suffer too much.

Online routing of FTP requests. In [GHPT99] we developed an algorithm for routing and scheduling large file transfers (e.g. images) in a general topology communication network. The requests (to transfer a file) arrive online and the goal is to eventually satisfy all the requests. Since the bandwidth of the links in the network is limited, it makes sense to try to schedule the transmissions in a way that utilizes the available resources optimally.

We considered the *online ftp problem*, which is a formal abstraction of the above file transfer problem. We assume that each ftp request specifies source/destination nodes and the size of

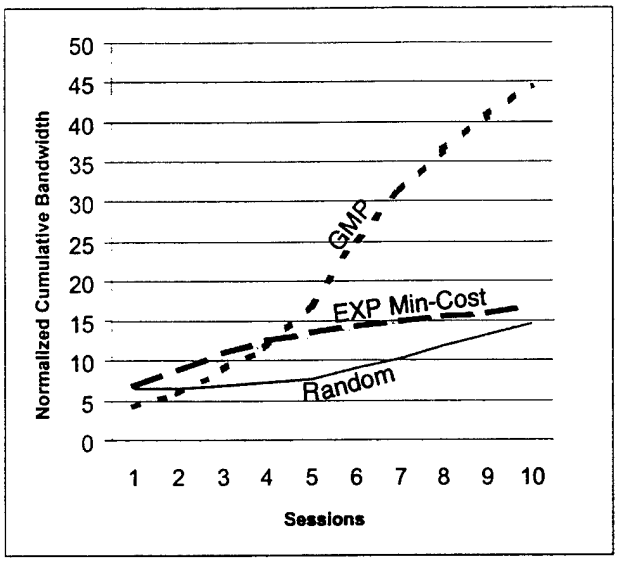

Figure 4: Comparing optimum fairness to combined fairness/throughput algorithm

the file. The goal of the online algorithm is to choose a path that will be used for transmitting each file, and to decide on the transmission rate. The main difference between this model and the (well-studied) models for online routing and admission control [GG92, AAF+93, GGK+93, AAP93] is that here we do not assume that the sources have prespecified transmission rate requirements, i.e. we can deal with non-streaming types of information.

Let n be the number of requests and P the maximum ratio between the sizes of the files. Assume that the smallest request can be processed in one time unit. Let F^*_{MAX} denote the optimum max-flow i.e. the smallest value for the maximum time a request spends in the system. Our main results are algorithms that achieve the *optimum max-stretch* and the *optimum total flow time* using resource augmentation. For the max-stretch algorithm we need to increase capacities by a factor of $O(\log P)$, whereas the total flow time algorithm needs a factor of $O(\log F^*_{MAX})$ increased capacity. The latter algorithm does not only achieve the optimum total flow time, but *simultaneously* optimizes many other objective functions, like the maximum flow time, the total square-of-flow-time, etc.

To justify the need for giving larger capacities to the online algorithm (i.e. resource augmentation), we showed polynomial lower bounds on both max-stretch and total flow time for the case where both online and offline algorithms are using the same capacities.

Facility Location and k-median Facility location is a classical optimization problem where we need to "buy" facilities in order to minimize total cost of the facilities plus the sum over all customers of the customer-to-facility distances. k-median is a variant of this problem where we are required to buy exactly k facilities. Publication [AGK+01] describes a local-search approach for approximately solving the k-median problem. The algorithm is quite practical

and the resulting approximation factor is the best currently known. Fault-tolerant version of the facility location problem is considered in [GMM01a]. Online version of the facility location problem, where the customers that need to be served appear online, is considered in [Mey]. Profit-earning variant, where each facility starts "bringing profit" if it serves enough customers, is considered in [Mey01].

Publication during the reporting period

[AGK+01] V. Araya, N. Garg, R. Khandekar, V. Pandit, A. Meyerson, and K. Munagala. Local search heuristics for k-median and facility location problems. In *Proc. 33rd Annual ACM Symposium on Theory of Computing*, June 2001.

[AM00] M. Andrews and K. Munagala. Online algorithms for caching multimedia streams. *Proceedings of the 8th European Symposium on Algorithms*, 2000.

[CCG+98] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin. Approximating a finite metric by a small number of tree metrics. In *Proc. 39th IEEE Annual Symposium on Foundations of Computer Science*, November 1998.

[GHP98] A. Goel, M. Henzinger, and S. Plotkin. Online throughput-competitive algorithm for multicast routing and admission control. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, January 1998.

[GHPT99] A. Goel, M. Henzinger, S. Plotkin, and E. Tardos. Scheduling data transfers in a network and the set scheduling problem. In *Proc. 31st Annual ACM Symposium on Theory of Computing*, May 1999.

[GMM00] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Hierarchical placement and network design problems. In *Proc. 41st IEEE Annual Symposium on Foundations of Computer Science*, November 2000.

[GMM01a] S. Guha, A. Meyerson, and K. Munagala. Improved algorithms for fault tolerant facility location. *Proceedings of the 12th ACM-SIAM SODA*, 2001.

[GMM01b] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Constant factor approximation for single sink edge installation problem. In *Proc. 33rd Annual ACM Symposium on Theory of Computing*, June 2001.

[GMP00] Ashish Goel, Adam Meyerson, and Serge Plotkin. Combining fairness with throughput: Online routing with multiple objectives. In *Proc. 32nd Annual ACM Symposium on Theory of Computing*, May 2000.

[GMP01] Ashish Goel, Adam Meyerson, and Serge Plotkin. Approximate majorization and fair online load balancing. In *Proc. 12th ACM-SIAM Symposium on Discrete Algorithms*, January 2001.

[GOPS98] A. Goldberg, J. D. Oldham, S. Plotkin, and C. Stein. An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow. In *Proc. IPCO-98*, June 1998.

[Mey] A. Meyerson. Online facility location. In *Proc. 42nd IEEE Annual Symposium on Foundations of Computer Science*, page 426.

[Mey01] Adam Meyerson. Profit earning facility location. In *Proc. 33rd Annual ACM Symposium on Theory of Computing*, June 2001.

[MMP00] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Cost-distance: Two metric network design. In *Proc. 41st IEEE Annual Symposium on Foundations of Computer Science*, November 2000.

[MMP01a] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Designing networks incrementally. In *Proc. 42nd IEEE Annual Symposium on Foundations of Computer Science*, pages 406–415, 2001.

[MG02] K. Munagala and A. Goel. Extending greedy multicast routing to delay sensitive applications. *Algorithmica*, 33, 2002.

[MMP01b] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Web caching using access statistics. In *Proc. 12th ACM-SIAM Symposium on Discrete Algorithms*, January 2001.

References

[AA97] B. Awerbuch and Y. Azar. Buy-at-bulk network design. *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 542–47, 1997.

[AAF+93] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line machine scheduling with applications to load balancing and virtual circuit routing. In *Proc. 25th Annual ACM Symposium on Theory of Computing*, pages 623–631, May 1993.

[AAK99] S. Albers, S. Arora, and S. Khanna. Page replacement for general caching problems. *Proceedings of 10th SODA*, 1999.

[AAP93] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput competitive on-line routing. In *Proc. 34th IEEE Annual Symposium on Foundations of Computer Science*, pages 32–40, November 1993.

[AGK+01] V. Araya, N. Garg, R. Khandekar, V. Pandit, A. Meyerson, and K. Munagala. Local search heuristics for k-median and facility location problems. In *Proc. 33rd Annual ACM Symposium on Theory of Computing*, June 2001.

[AM00] M. Andrews and K. Munagala. Online algorithms for caching multimedia streams. *Proceedings of the 8th European Symposium on Algorithms*, 2000.

[AMO96a] Y. Afek, Y. Mansour, and Z. Ostfeld. Convergence complexity of optimistic rate based flow control algorithms. *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 89–98, 1996.

[AMO96b] Y. Afek, Y. Mansour, and Z. Ostfeld. Phantom: A simple and effective flow control scheme. *ACM SIGCOMM*, pages 169–182, 1996.

[AS98] B. Awerbuch and Y. Shavitt. Converging to approximated max-min flow fairness in logarithmic time. *Proceedings of the 17th IEEE Infocom conference*, pages 1350–57, 1998.

[Bar96] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *37th IEEE symposium on Foundations of Computer Science*, pages 184–193, 1996.

[Bar98] Y. Bartal. On approximating arbitrary metrics by tree metrics. *30th ACM Symposium on Theory of Computing*, 1998.

[BBR97] Y. Bartal, J. Byers, and D. Raz. Global optimization using local information with applications to flow control. *38th Annual Symposium on Foundations of Computer Science*, pages 303–312, 1997.

[BCF+99] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. *Proceedings of INFOCOM*, 1999.

[BFCAZ99] Y. Bartal, M. Farach-Colton, M. Andrews, and L. Zhang. Fast fair and frugal bandwidth allocation in atm networks. *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 92–101, 1999.

[BG92] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall, 1992.

[CCG+98] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin. Approximating a finite metric by a small number of tree metrics. In *Proc. 39th IEEE Annual Symposium on Foundations of Computer Science*, November 1998.

[CK99] Edith Cohen and Haim Kaplan. Exploiting regularities in web traffic patterns for cache replacement. *Proceedings of 31st ACM STOC*, 1999.

[CN96] J. Castro and N. Nabona. An implementation of linear and nonlinear multicommodity network flows. *European Journal of Operational Research*, 92(1):37–53, July 1996.

[DS96] A. Dan and D. Sitaram. A generalized interval caching policy for mixed interactive and long video environments. *Multimedia Computing and Networking*, January 1996.

[FKL+91] A. Fiat, R. Karp, M. Luby, L. McGeoch, D. Sleator, and N. Young. Competitive paging algorithms. *Journal of Algorithms*, 12:685–699, 1991.

[GG92] J. Garay and I. Gopal. Call preemption in communication networks. *IEEE INFOCOM*, pages 1043–1050, 1992.

[GGK+93] J. Garay, I. Gopal, S. Kutten, Y. Mansour, and M. Yung. Efficient on-line call control algorithms. *2nd Annual Israel Conference on Theory of Computing and Systems*, 1993.

[GHP98] A. Goel, M. Henzinger, and S. Plotkin. Online throughput-competitive algorithm for multicast routing and admission control. In *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms*, January 1998.

[GHPT99] A. Goel, M. Henzinger, S. Plotkin, and E. Tardos. Scheduling data transfers in a network and the set scheduling problem. In *Proc. 31st Annual ACM Symposium on Theory of Computing*, May 1999.

[GK95] Michael D. Grigoriadis and Leonid G. Khachiyan. Approximate minimum-cost multicommodity flows in $\tilde{O}(\epsilon^{-2}knm)$ time. Technical Report 95-13, Rutgers University, May 1995.

[GMM00] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Hierarchical placement and network design problems. In *Proc. 41st IEEE Annual Symposium on Foundations of Computer Science*, November 2000.

[GMM01a] S. Guha, A. Meyerson, and K. Munagala. Improved algorithms for fault tolerant facility location. *Proceedings of the 12th ACM-SIAM SODA*, 2001.

[GMM01b] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Constant factor approximation for single sink edge installation problem. In *Proc. 33rd Annual ACM Symposium on Theory of Computing*, June 2001.

[GMP00] Ashish Goel, Adam Meyerson, and Serge Plotkin. Combining fairness with throughput: Online routing with multiple objectives. In *Proc. 32nd Annual ACM Symposium on Theory of Computing*, May 2000.

[GMP01] Ashish Goel, Adam Meyerson, and Serge Plotkin. Approximate majorization and fair online load balancing. In *Proc. 12th ACM-SIAM Symposium on Discrete Algorithms*, January 2001.

[GOPS97] A. Goldberg, J. D. Oldham, S. Plotkin, and C. Stein. An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow. Technical Report STAN-CS-TR-97-1600, Stanford University, December 1997.

[GOPS98] A. Goldberg, J. D. Oldham, S. Plotkin, and C. Stein. An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow. In *Proc. IPCO-98*. June 1998.

[HLP29] G.H. Hardy, J.E. Littlewood, and G. Polya. Some simple inequalities satisfied by convex functions. *Messenger Math.*, 58:145–152, 1929.

[HLP52] G.H. Hardy, J.E. Littlewood, and G. Polya. *Inequalities*. 1st ed., 2nd ed. Cambridge University Press, London and New York., 1934, 1952.

[HNG+99] M. Hofmann, T.S.E. Ng, K. Guo, S. Paul, and H. Zhang. Caching techniques for streaming multimedia over the internet. *Bell Laboratories Technical Memorandum*, 1999.

[Ira97] S. Irani. Page replacement with multi-sized pages and applications to web caching. *Proceedings of 29th ACM STOC*, 1997.

[KK00] A. Kumar and J. Kleinberg. Fairness measures for resource allocation. In *Proc. 41st IEEE Annual Symposium on Foundations of Computer Science*, November 2000.

[KP95] David Karger and Serge Plotkin. Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows. In *Symposium on the Theory of Computing*, volume 27, pages 18–25. Association for Computing Machinery, ACM Press. May 1995.

[KPR99] M. Korupolu, G. Plaxton, and R. Rajaraman. Placement algorithms for hierarchical cooperative caching. *Proceedings of the 10th ACM-SIAM SODA*, 1999.

[KRT99] J. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 1999.

[Mey] A. Meyerson. Online facility location. In *Proc. 42nd IEEE Annual Symposium on Foundations of Computer Science*, page 426.

[Mey01] Adam Meyerson. Profit earning facility location. In *Proc. 33rd Annual ACM Symposium on Theory of Computing*, June 2001.

[MG02] K. Munagala and A. Goel. Extending greedy multicast routing to delay sensitive applications. *Algorithmica*, 33. 2002.

[MMP00] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Cost-distance: Two metric network design. In *Proc. 41st IEEE Annual Symposium on Foundations of Computer Science*, November 2000.

[MMP01a] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Designing networks incrementally. In *Proc. 42nd IEEE Annual Symposium on Foundations of Computer Science*, pages 406–415, 2001.

[MMP01b] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Web caching using access statistics. In *Proc. 12th ACM-SIAM Symposium on Discrete Algorithms*, January 2001.

[MS91] L. McGeoch and D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6:816–825, 1991.

[PST95] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, May 1995.

[SCRS97] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Buy-at-bulk network design: Approximating the single-sink edge installation problem. *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 619–628, 1997.

[ST85] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[TRS97] I. Tatarinov, A. Rousskov, and V. Soloviev. Static caching of web servers. *Proceedings of the Sixth International Conference on Computer Communications and Networks*, 1997.

[ZIRO99] J. Zhang, R. Izmailov, D. Reininger, and M. Ott. Web caching framework: Analytical models and beyond. *IEEE workshop on Internet Applications*, 1999.